云环境下面向负载均衡的数据密集型工作流的数据约简策略 *

胡志刚，李　佳，郑美光

(中南大学 软件工程学院, 长沙 410075)

摘　要：如何对数据密集型工作流应用进行高效合理地调度成为云计算领域亟待解决的关键问题之一。针对此问题，首先构造数据密集型工作流的有向超图模型。然后提出了数据支持能力概念，通过基于数据支持能力的合并操作对模型进行约简。最后优化超图多层剖分算法，提出数据约简的数据密集型工作流调度策略 HEFT-P。研究结果表明，HEFT-P 相比典型的工作流调度策略 HEFT、CPOP、MCP，能够很好地对数据密集型工作流进行约简优化，获得较少的调度时间。

关键词：数据密集型工作流；有向超图；数据约简调度；云计算；负载均衡

中图分类号：TP391　　doi: 10.3969/j.issn.1001-3695.2018.02.0143

Data reduced strategy for load-balanced data-intensive workflow in clouds

Hu Zhigang, Li Jia, Zheng Meiguang

(*College of Software Engineering*, *Central South University*, *Changsha* 410075, *China*)

Abstract: How to schedule data-intensive workflow efficiently and reasonable has become one of the key issue in cloud computing. To address this issue, first, we build a directed hypergraph model for data-intensive workflow. And proposing a concept data supportive ability to help the presentation of data-intensive workflow application and provide the merge operation details considering the data supportive ability. By optimizing the hypergraph multi-level partitioning algorithm, proposing a data reduced scheduling policy HEFT-P for data-intensive workflow. Through simulation, the classical HEFT, CPOP and MCP scheduling policies was compared with HEFT-P. The results indicate that HEFT-P could obtain reduced data scheduling and reduce the makespan of executing data-intensive workflows.

Key words: data-intensive workflow; directed hypergraph; data reduced scheduling; cloud computing; load balancing

chinaXiv:201805.00481v1

0　引言

数据密集型计算已经发展成为一种充满前景的新范式，它旨在通过分析研究大量数据来更好地理解问题并细化问题的求解区域[1]。云计算的按需访问，在虚拟计算环境下运行大规模应用的可伸缩性和可用性，为大数据的数据存储、管理与分析等方面提供了重要支撑[2]。基于云计算的大型网络应用呈现出分布、异构的特点和数据密集的趋势，如科学工作流系统，这类应用被称为数据密集型应用(Data-Intensive Applications)[3]，这类应用的数据密集型主要体现在其处理的数据大小通常达 TB 甚至 PB 级。

数据密集型应用中的任务往往要获取、处理和传输大量的数据，不合理的数据选择和任务调度策略会导致数据的传输量和访问量过大。不但增加了用户使用云资源的费用，还严重影响了科学工作流的执行效率。因此，如何对数据密集型工作流进行约简优化，简化大数据的表征，实现大数据按需约简、降低复杂度以获取更好的知识抽象，在大数据应用中成为了一个非常有价值的研究课题。本文的工作目的是对云环境下数据密集型应用进行工作流数据约简及任务调度，通过减少数据的反复传输及任务间的通信量来减少工作流的调度时间。

准确的任务调度模型是对问题进行研究的基础，它有助于研究人员对问题的理解和分析，从而更好地解决问题。目前表示依赖关系的任务调度模型表示主要有 task priority graph (TPG) 和 directed acyclic graph (DAG)。在 DAG 图中每个顶点代表应用中的一个任务，每条边代表任务间的数据依赖。但随着云环境下的数据密集型应用的出现，任务间的关系变得更加复杂，为了能够更加准确地表示任务间的真实关系，本文采用有向超图对数据密集型工作流进行表示。有向超图的一条超边可以连接多个节点，能够很好地表示非对称的依赖关系[4,5]，清晰地表述任务间的多对多关系[6]。它除了拥有一般图论所具有的概念之外，还具有超边合并、分割等一般图所不具备的能力。超图分割作为图论中的一个典型组合优化问题，在超大规模集成电

收稿日期：2018-02-12；**修回日期：**2018-05-08　**基金项目：**国家自然科学基金资助项目 61602525，61572525）

作者简介：胡志刚（1963-），男，山西吕梁人，教授，博士，主要研究方向为网络并行处理、嵌入式系统、软件工程（zghu@csu.edu.cn）；李佳（1991-），女，硕士研究生，主要研究方向为云计算、大数据；郑美光（1983-），女，副教授，硕导，博士，主要研究方向为云计算、大数据.

ChinaXiv合作期刊

路设计[7]、并行计算[8]、图像识别等领域都有着广泛的应用。正是由于超图分割的优越性，国内外学术界提出了大量优秀的分割方法，主要有多层算法、谱方法[9]、KL/FM(Kernighan Lin/Fiduccia Mattheyses)[10]以及各种局部优化算法等。而包含这些算法的图分割软件包也相继出现，如 hMETIS、PaToH 等，Devine 等人[11]也提出了并行的分割软件 Zoltan toolkit，实现了对超图更加高效的分割。一些学者利用超图的以上优点来解决实际问题，孙雪冬等人[12]提出了基于有向超图的工作流资源分配均衡优化方法，利用超图的性质，以及活动能力需求集与过程结构相互作用的形式化规则来对企业过程进行优化。Laura 等人[13]采用有向超图对学习活动和与其相关联的组件能力的保持关系进行建模，对相关算法来进行调整并构建了学习组件的大规模资源库。

目前，数据密集型应用已被广泛应用于天文学、高能物理学等领域，如何对它们进行高效合理的调度成为云计算领域的一个研究热点。经典的依赖任务调度算法heterogeneous earliest finish time (HEFT)和critical path on a processor (CPOP)，根据任务优先级对任务进行选择，并将其分配到合适的处理器，取得了较短的调度时间。但这两种算法在处理器数目过多和一些简单的情况下不能获得比较好的结果。肖鹏等人[14]通过在传统DAG图中引入虚拟数据存储节点，提出了面向数据密集型工作流的能耗感知的调度策略，从能耗感知的角度来对数据密集型工作流中任务进行调度，从而降低任务执行能耗。苑迎春等人[15]通过分析DAG图中活动的并行和同步完成特征，将工作流截止期转化为层截止期，提出了截止日期约束的逆向分层费用优化算法，在时间约束内对成本进行优化。Ahmad等人错误!未找到引用源。提出混合遗传算法(hybrid genetic algorithm,HGA)，通过在初始种群中引入启发式策略HEFT为最优解的寻找提供方向，能够在较短时间内快速收敛到最优解，但该算法时间复杂度较高。Lin等人[17]提出多云环境下带截止日期约束的代价驱动工作流调度策略，该策略引入局部关键路径理论，对工作流的子任务进行局部整体分配，达到压缩数据通信、减少执行代价的目的，然而该策略忽略了云间的数据通信代价和实例启停时间等因素。Mon等人错误!未找到引用源。提出了云环境下基于任务依赖关系的任务聚类方（CMTD），以减少执行开销并提高科学工作流任务的计算粒度，但是该算法考虑的计算资源并非异构情况，不符合真实的云计算环境。

综上分析，本文提出云环境下基于超图的数据约简调度策略，实现减少计算节点间通信量和工作流完成时间的目的。总体思路与方法如下：

a）根据数据密集型应用的特性对数据文件进行抽象，提出数据支持能力的概念，构建考虑了数据支持能力的有向超图模型，更好地表达任务间的关系；

b）根据数据支持能力合并任务，优化超图粗化策略，约简并剖分模型从而最小化计算节点间的通信量；

c）在此基础上提出任务调度算法 HEFT-P，每个任务子集的总指令长度与所在计算节点的指令执行速度成比例，使工作流在满足负载均衡约束下获得更短的完成时间。

1　数据密集型工作流的有向超图模型

1.1　模型定义

在一个数据密集型应用中，两个任务的问题求解可能不相同，但满足它们处理所需要的数据元素却可能存储在相同的数据文件中。这就必然导致同一个文件可能被多个任务请求，并且传输到任务所在的计算节点。本节利用向超图属性与数据密集型工作流属性间的对应关系，根据文件与任务之间的关系构建模型。

一个数据密集型工作流的有向超图模型 $\vec{H}=(V,E)$，其中 V 是超图的所有节点集合，它分为两类节点集合，$V=D\cup T$，其中 D 代表数据集节点集合，T 代表任务节点集合，$D\cap T=\varnothing$。E 为超图的超边集合，它也分为两类边集，$E=X\cup Y$，X 为数据对任务的支持边集合，Y 为任务间依赖的工作流超边集合，$X\cap Y=\varnothing$。w 表示节点的权重，c 代表超边的权重。

对于任务集合 $T=\{t_1,t_2,\ldots,t_n\}$，t_i 为数据密集型应用中的任务节点，记 $pred(t_i)$ 为 t_i 的前驱节点集，$succ(t_i)$ 为 t_i 的后继结点集。没有前驱的 t_i 称为入口节点，记作 t_{entry}，没有后继的 t_i 称为出口节点，记做 t_{exit}。在此基础上，本文定义工作流超边。

定义 1　工作流超边 Y。$\forall y_m\in Y$，由当前任务 t_i 指向它的后继任务 t_j，即 $\exists t_i$，$t_j\subseteq T$，$t_i\in T(y_m)$，$t_j\in H(y_m)$。其中，$T(y_m)$ 为超边 y_m 的尾节点集合，$H(y_m)$ 为超边 y_m 的头结点集合。$\forall t_i$，$t_j\in T$，若 $\exists y_m\in Y$，有 $t_i=\mathrm{T}(y_m)$，$t_j=H(y_m)$，称 t_i 与 t_j 为可操作相邻任务，记做 $\wedge(t_i,t_j)$。

为了高效地为任务选择合适的数据，更好地对数据能够实现的功能进行分析，获取数据知识抽象，本文增加了数据支持能力的标识，定义如下：

定义 2　数据支持能力 a_k。在数据密集型应用内，通过 NWS 服务[18]获取输入数据所能实现的历史功能，将其结果记作数据支持能力 a_k，并将此标志赋予其对应的输入文件。其中，k 为数据支持能力的编号，即数据所能实现的功能类型。应用内所有种类的数据支持能力共同构成能力类型集合 A，即 $A=\{a_1,a_2,\ldots,a_p\}$，其中，p 为数据支持能力类型集合所具有的能力类型总数。

对于数据密集型应用中的输入文件集合 $D=\{d_1,d_2,\ldots,d_m\}$，d_j 为数据集文件，下文中称作数据集节点。基于此，本文定义数据支持超边。

定义 3　数据支持超边 X。$\forall x_m\in X$，由 d_j 指向以其作为输入数据的 t_i，即 $d_j\in D$，$t_i\in T$，$d_j\in T(x_m)$，$t_i\in H(x_m)$，其中，$T(x_m)\subseteq D$，$H(x_m)\subseteq T$。

d_j 可以和 A 中的数据支持能力进行映射，构成 d_j 所具有的数据支持能力集 S_j，所有 d_j 的 S_j 与 A 共同构成了 $m\times p$ 的数据集支持能力矩阵(Dataset Supportive Ability Matrix, *DM*)，其

chinaXiv:201805.00481v1

ChinaXiv合作期刊

中 m 为数据集节点的个数。t_i 与 A 也存在映射关系，它构成了 t_i 执行时的数据支持能力需求集 R_i 。所有 t_i 的 R_i 与 A 共同构成 $p\times n$ 的任务需求能力矩阵(Task Requirement Ability Martix, *TM*)，n 为任务的个数。本文通过集成 *DM* 与 *TM* 构造一个 $m\times p\times n$ 的三维数据集-任务能力矩阵(Dataset-Task Ability Matrix, *DTAM*)，如图 1 所示。如，$\Gamma(d_j,a_k,t_i)=1$，$1\le j\le m$，$1\le k\le p$，$1\le i\le n$，表示 d_j 能够提供给 t_i 执行时所需的支持能力 a_k 。

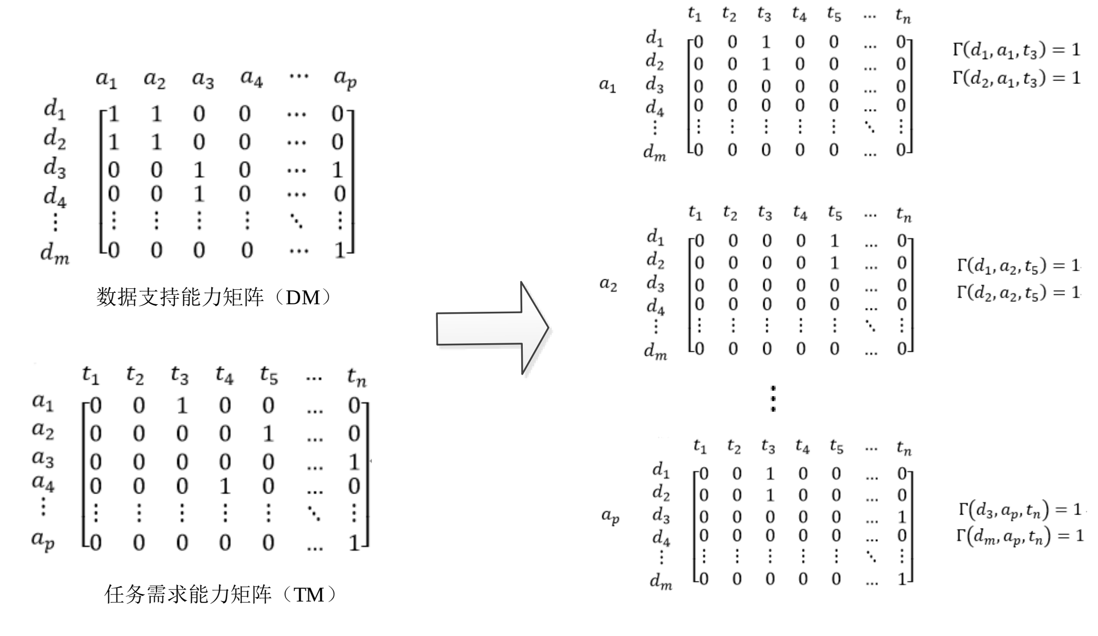


图 1　数据集-任务能力矩阵 DTAM

基于以上定义，对有向超图模型的节点进行表述。

任务节点 $t_i=(i,R_i,I,O)\in T$，其中，i 为任务节点的编码，$R_i\in A$。I 为 t_i 执行时的输入数据，O 为 t_i 执行完成时的输出数据。t_i 的支持数据集记作：S_i^{d}；数据集节点 $d_j=(j,S_j,\Lambda)\in D$，其中，j 为数据集节点的编码，$S_j\subset A$。$\forall x_n\in X$，若 $t_i\in H(x_n)$，$d_j\in T(x_n)$，则 $\{R_i\in S_j|R_i\in A,S_j\subset A\}$。

以图 2 为例，数据集节点 d_1、d_2 具有的支持能力为：$S_1=\{a_1,a_2\}$，$S_2=\{a_1,a_2\}$，任务 T 所需要的数据能力分别为：$R_3=\{a_1\}$，$R_5=\{a_2\}$。由矩阵 *DTAM* 可得：$\Gamma(d_1,a_1,t_3)=1$，$\Gamma(d_2,a_1,t_3)=1$，$\Gamma(d_1,a_2,t_5)=1$，$\Gamma(d_2,a_2,t_5)=1$。则 t_3、t_5 的支持数据集分别为 $S_3^{d}=\{d_1,d_2\}$，$S_5^{d}=\{d_1,d_2\}$。本文考虑了数据密集型工作流中任务、数据以及数据支持能力三者的关系，根据 *DTAM* 矩阵得到了图 2，考虑了数据支持能力的数据密集型工作流的有向超图模型，图中圆形代表任务，六边形代表输文件。通过该映射操作，本文将数据密集型工作流的调度问题转化为有向超图的数据约简及分割问题。通过该映射操作，本文将数据密集型工作流的调度问题转化为有向超图的数据约简及分割问题。

1.2　问题描述

根据上述对于数据密集型工作流的模型定义，数据密集型工作流的调度时间 *makespan* 由超图模型中的关键路径(*critical path, CP*)所决定，即超图中最长的路径所决定。*makespan* 的计算采用文献[4]中，如下所示：

$$makespan=\max_{t_i\in V}\{tlevel(t_i)+blevel(t_i)\} \tag{1}$$

其中，$tlevel(t_i)$ 为从入口节点 t_{entry} 到 t_i 的最长路径，即从入口任务到当前任务开始执行所需要的时间，其中不包括 t_i 的权重。$blevel(t_i)$ 为从 t_i 到出口节点 t_{exit} 的最长路径，它们的计算公式为，

$$blevel(t_i)=w_i+\max_{t_j\in succ(t_i)}\{c_{i,j}+blevel(t_j)\} \tag{2}$$

$$tlevel(t_i)=\max_{t_j\in pred(t_i)}\{c_{i,j}+tlevel(t_j)\} \tag{3}$$

其中：w_i 为任务 t_i 的权重，即任务的计算量，$c_{i,j}$ 为任务 t_i 与其后继任务的通信量。由以上公式可见，关键路径上关键任务的计算量没有办法改变，本文将通过对关键任务进行合并或者将其分配在同一计算节点增强本地化等操作来减少任务间的通信量，从而减少任务的调度时间 *makespan* 。

在对数据密集型工作流的超图模型进行操作时，将采用超图分割理论作为主要目标指导。给定一个有向超图 $\vec{H}=(V,E)$ 和一个整数 k，将 T 分割为 k 个权重近似的节点子集，$Partition=\{p_1,p_2,\ldots,p_k\}$，其中，$p_l$ 的权重为 $w_l^{p}=\sum_{t_i\in p_l}w_i$。它们需要满足以下条件：a) $p_i\cap p_j=\varnothing,(i,j=1,2,\ldots k)$；b) $\bigcup_{l=1}^{k}p_l=T$；c) $W_{avg}=\frac{\sum_{t_i\in V}w_i}{k},w_l^{p}\le W_{avg}(1+\varepsilon)$。其中：$\varepsilon$ 为预先定义的负载均衡的阈值。如果属于 y_i 的节点至少属于两个不同的分区 p_i、p_j 时，称 y_i 被切割，本文尝试最小化连通度，即被切割的边数，

chinaXiv:201805.00481v1

ChinaXiv合作期刊

$$cuts(\overline{H},P)=\sum_{i=0}^{|Y|-1}(\lambda_i(\overline{H},P)-1) \tag{4}$$

其中，$\lambda_i(\vec{H},P)$ 为超边 y_i 所连接的分区数。$cuts(\overline{H},P)$ 能够准确地反映出并行计算中的通信开销，最小化连通度对应于负载均衡条件下不同虚拟机间通信量最小的目标。

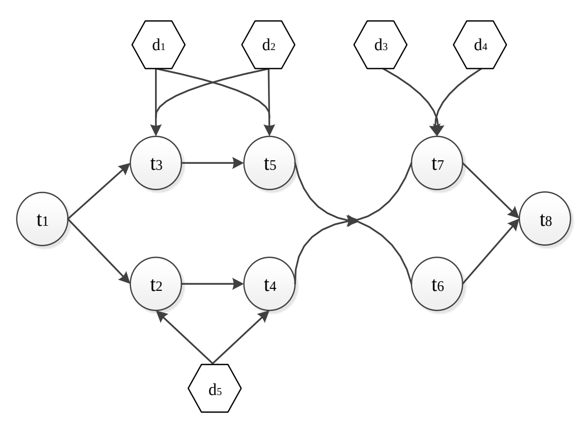


图 2　考虑支持能力的有向超图模型

2　数据约简的超图分割调度算法

本节将通过基于支持能力的数据约简与超图切割算法来对有向超图模型进行结构优化，减少任务的总通信量，从而减少任务的调度时间。基于支持能力的数据约简将通过对任务的合并操作来实现，从而减少相同数据的反复传输。超图分割是一个 NP-hard 问题，可采用基于启发式的方法来有效解决这个问题，文中将采用多层算法对数据密集型工作流超图模型进行分割，并对分割算法进行优化，以获得更好的性能。

2.1　基于支持能力的合并

在数据密集型工作流中，任务的输入输出数据量往往非常大，但这其中存在一些相邻任务，它们拥有共同支持数据，如果将它们分配到同一个虚拟机上，则可以减少因等待数据传输而造成的时间消耗。本文将基于任务与支持能力的关系根据 *DTAM* 矩阵求得任务的支持数据集，并按照以下合并规则对这些任务进行数据约简，减少相同支持数据的反复传输，从而减少任务的执行时间，降低算法的复杂度。

规则 1　合并处理。$\forall\wedge(t_i,t_j)$，有 $S_i^d\cap S_j^d\neq\varnothing$，则任务 t_i 与 t_j 可进行合并操作。若合并后的任务为 t_c，则 $t_c.I=t_i.I\cup t_j.I-t_i.O\cap t_j.I$，$t_c.O=t_i.O\cup t_j.O-t_i.O\cap t_j.I$，$S_c^d=S_i^d\cap S_j^d$。

本文将基于以上合并处理的规则，采用任务合并算法(SMA)对任务进行合并操作，SMA 描述如下：

算法 1　任务合并算法(SMA)

输入： $DTAM$，$\vec{H}=(V,E)$

输出：$\overrightarrow{H_{init}}=(V_{init},E_{init})$

a) begin

b) for $t_i\in T$ and $R_i\in A$ do //求任务的支持数据集

c) $a_i\leftarrow R_i,S_i^d\leftarrow\{\}$ //将 R_i 赋给 a_i，S_i^d 设为空

d) for $d_j\in D$ do //对于每一个数据集节点

e) if $\Gamma(d_j,a_i,t_i)=1$ then

f) $S_i^d\leftarrow S_i^d\cup d_j$

g) end if

h) end for

i) end for

j) for $t_i\in T$ do //判断相邻任务是否需要进行合并

k) if $\wedge(t_i,t_j)$ then

l) if $S_i^d\cap S_j^d\neq\varnothing$ and $w_i+w_j<W_{avg}(1+\varepsilon)$ then

m) $Combine(t_i,t_j)$

n) end if

o) end if

p) end for

q) end

伪代码第 2~9 行求工作流中任务的支持数据集，将任务 i 的支持数据集 S_i^d 设为空，每个任务所需的 R_i 赋给 a_i，将满足 $\Gamma(d_j,a_i,t_i)=1$ 的数据集节点，并入集合 S_i^d；第 10-16 行判断任务是否需要合并，若 t_i 与 t_j 为可操作相邻任务，支持数据集的交集不为空，并且两任务的权重之和小于预定的最大负载，则根据规则 1 合并 t_i 与 t_j。合并后的节点权重为所合并的节点权重之和。算法的时间复杂度为 $O(n\times m)$，n 为任务数目，m 为数据文件的数目。

2.2　面向支持数据的多层剖分算法

在采用 SMA 算法对任务合并后，本文将采用多层剖分算法对超图模型进行分割，达到最小化计算节点间通信量的目的。多层剖分算法一般分为粗化、分割、细化三个阶段，在粗化阶段，迭代进行节点匹配与合并操作直至满足终止条件，并构建多水平的粗化超图，有效地减少了原始超图中超边的条数和节点的数目；在分割阶段，对粗化阶段得到的最小粗化超图进行初始划分；在细化阶段，将最小粗化超图的初始剖分投影回原始超图，并在每一层的投影中对超图划分进行优化，有效地解决迁移原始超图中节点组的问题。本节对多层剖分算法的粗化阶段进行优化。

2.2.1 基于支持数据的超图粗化

对节点进行合并处理之后，将对有向超图进行粗化操作。在粗化阶段，原始的有向超图 $\vec{H}=(V,E)$ 将会由一个个越来越小的有向超图 $\overrightarrow{H_i}=(V_i,E_i)$ 来表示，即 $|V_i|<|V_{i-1}|<\ldots<|V_{init}|$，$|V|$ 为节点的数目。其中，$\overrightarrow{H_i}$ 中的节点通过不同的匹配策略进行匹配、合并来构成下一个更小的粗化超图 $\overrightarrow{H_{i+1}}$。在 $\overrightarrow{H_i}$ 水平匹配的两个节点将会在 $\overrightarrow{H_{i+1}}$ 阶段合并为一个节点，合并后的节点权重为所合并的节点权重之和。

然而超图粗化策略的选取对粗化剖分的质量至关重要，现有的贪婪匹配算法主要考虑超边权重，尽可能多地对权重较大的超边中的节点进行匹配，并没有考虑节点权重对匹配效果及粗划剖分结果的影响。为得到更好的粗化效果，本文在此基础

chinaXiv:201805.00481v1

ChinaXiv合作期刊

chinaXiv:201805.00481v1

上通过考虑节点及超边权重对粗化效果的影响，优化节点匹配策略。提出的基于数据支持能力的粗化策略（CBSD）流程如图 3 所示。

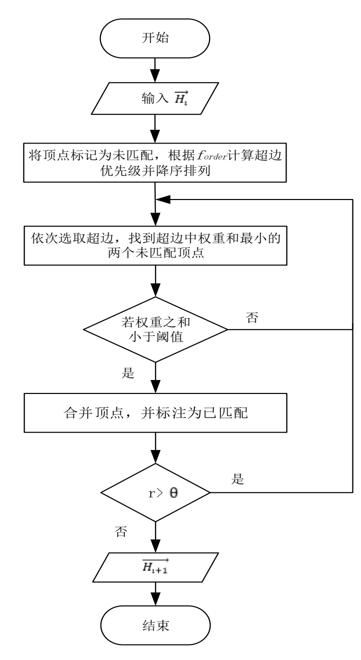


图 3 基于数据支持能力的粗化算法(CBSD)流程图

首先，本文将采用排序函数 f_{order} 对任务超边进行降序排列；按照顺序依次对超边中的节点进行匹配，这样便可对节点权重较小但其所属的具有较大边权的超边进行优先匹配，一方面尽可能多地隐藏边权，另一方面又能避免合并后的节点权重过大，有利于计算初始粗化剖分和提高细化算法的效率。基于上面的分析，排序函数 f_{order} 定义如下：

$$f_{order}\left(y_m\right)=c\left(y_m\right)\sum_{t_i\in y_m}\frac{1}{w_i}\quad(y_m\in Y)\tag{5}$$

当节点削减率 r 满足终止条件 θ，该轮粗化操作终止，通常当 r 的值小于 10%时终止粗化操作。

2.2.2 粗化分割及细化阶段

当超图足够小，即削减率小于阈值时，进入粗化分割阶段，这一阶段的目的是，通过对上一阶段得到的最小可用超图进行分割来得到一个原始超图的初始分割结果。然而，将粗化图分割为 k 份并不能代表原始图的高质量分区，但细化阶段中重复的本地优化操作能够很好地解决这个问题。细化阶段的主要目的是在满足虚拟机间负载均衡的条件下降低分区间的连通度 $cuts(\vec{H},P)$。选取粗化阶段中的最优分割结果，细化阶段通过多次迭代操作将其映射回原始超图，并在每次迭代中使用本地优化算法来提高它质量。其中较成功的细化方法主要是基于 Kernighan-Lin(KL)和 Fiduccia- Mattheyses (FM)启发式算法的优化与推广。

2.3 调度算法

文中基于 HEFT 算法提出了 HEFT-PS 调度算法，以上节分割后得到的分区 p_l 为单位对任务进行调度，从而实现对数据密集型工作流的高效调度。任务优先级 $rank_u(t_i)$ 的计算如式(6)所示。传统算法将会直接根据任务的优先级对单个任务进行调度，而本文提出的 HEFT-PS 调度算法则将会以任务集合为单位对任务进行调度。令 t_i 为 p_l 中优先级最大的任务，为满足任务执行的依赖关系，该算法将 p_l 的优先级设为 t_i 的优先级，以尽快对已经准备好的任务进行调度，从而降低该任务的等待时间。即将 p_l 所包含的任务中最大的 $rank_u(t_i)$ 值作为 p_l 的优先级 $rank_i$，

$$rank_i=\max_{t_i\in p_l}\left\{rank_u\left(t_i\right)\right\}\tag{6}$$

文中将会根据优先级 $rank_i$ 对 p_l 进行调度。经过 HEFT-PS 调度算法调度之后，每个 p_l 都被分配到了合适的虚拟机上。HEFT-PS 算法描述如下：

算法 3 HEFT-PS 调度算法

输入： *Partition*

输出： 任务执行结果

1) begin
2) 计算所有任务的 $rank_u$，及所有 *Partition* 的 $rank_i$
3) 将 p_i 在调度列表里按照 $rank_i$ 的非递增顺序进行排序
4) while 列表中有未调度 p_l
5) 　　从调度列表里取第一个 p_l
6) 　　for 每个计算节点 $v_m\in Q$
7) 　　　　计算 p_l 在 v_m 上的执行时间 $EFT\left(p_l,q_k\right)$
8) 　　end for
9) 　　将 p_l 放在 $EFT\left(p_l,v_m\right)$ 最小的计算节点上
10) end while
11) end

伪代码第 2、3 行计算每一个分区 p_l，即任务集合优先级并按照非递增顺序放入调度列表中；第 4-10 行依次选取未调度的 p_l，计算其在每一个计算节点上的 EFT，并将其调度至具有最小 EFT 的计算节点上。调度结束后，将会根据每个任务的 $rank_u$ 在其所在的计算节点上顺序执行。该算法的时间复杂度为 $\boldsymbol{O}(n*p^2)$，p 为计算节点的个数。HEFT、CPOP 算法的时间复杂度为 $\boldsymbol{O}(n^2*p)$，MCP 的时间复杂度为 $\boldsymbol{O}(n^2*\log n)$，其中，$n$ 为任务的个数。p 的数值相对 n 而言一般较小，在 p 值相同的条件下，随着 n 的增加，HEFT-PS 算法的优势将会越来越明显。

3 实验

本文选用墨尔本大学网格实验室的 CloudSim 云计算仿真平台[错误!未找到引用源。]进行实验，为验证 HEFT-P 调度算法的性能，数据密集型工作流应用 DAG 图分别通过模拟设置和实际工作流应用两种方式产生。

本文开发了数据密集型应用(data-intensive application, DIA)集成测试平台，能够自动化定制生成数据密集型应用、验证数据密集型应用是否满足数据密集型特征，以及在验证数据密集型应用满足数据密集型特征后，存储所生成的数据密集型应用。IOzone [错误!未找到引用源。]是一个经典的数据密集型任务和文件系统的

ChinaXiv合作期刊

测试工具，该工具主要用以测试文件系统的读写性能。每一个 iozone 操作都是独立的测试任务，具有原子性。平台基于 iozone 中三个核心原子操作(R,W,RW)自动组合定制生成了一种的典型数据密集型应用 Iozones。用户可输入 IOZones 的主要参数，最大节点数，最小节点数，DAG 图的层数，最大宽度以及 RW 任务的个数对 IOZones 应用进行定制。生成的数据密集型应用以 DAG 图的形式保存在本地，会产生两个文件，分别保存 DIA 各个节点之间依赖关系的 DAG 图信息文件 iozones_dag.txt，以及保存每个节点中的命令信息 iozones_command.txt。

实际工作流采用 CyberShake 及 Montage 标准科学工作流 [1]。模拟设置任务数目分别为{50,100}和{25,100}个，任务数目反映了工作流应用规模。工作流实例选自于(数据密集型应用集成测试平台)项目实例库中的 benchmark 实例。任务平均通信时间与平均执行时间的比值 CCR 表示任务类型的集合，当 CCR>1 时，表示任务集合的通信比重大于执行比重，为数据密集型任务，实验中设置 CCR>1。每种规模生成 30 组不同的实例。模拟设置计算节点数目均为{4,8,16,32,64}个，负载均衡阈值为{10%,20%,30%,40%, 49%}。综合以上的问题参数，共生成(3*30＋2*2)*5*5=2350 个测试实例。实验采用 HEFT、CPOP 和 MCP 经典调度算法与本文调度算法进行实验结果比较，效果评估的性能指标为平均调度长度(Average Schedule Length, ASL)及平均资源利用率。为了保证结果的可靠性，取运行 500 次得到的 ASL 平均值作为测试结果，实验数据均位于其 95%的置信区间内。

定义 4　平均资源利用率(average resource utilization ration, ARUR). 平均资源利用率的计算公式如下：

$$ARUR = \frac{\overline{V_{FT}}}{makespan} \times 100\% \tag{7}$$

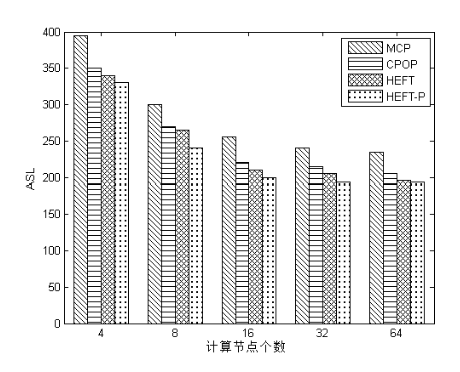
其中：$\overline{V_{FT}}$ 为计算节点上的平均负载，定义为总任务的最优完成时间，即任务的总指令长度除以计算节点指令执行速度之和。ARUR 的范围为[0,1]，ARUR 越接近于 1，说明调度算法的负载均衡性能越好。为了保证结果的可靠性，取运行 500 次得到的 ASL 平均值作为测试结果，实验数据均位于其 95%的置信区间内。

实验的硬件环境为 Intel(R) Core™ i5 CPU 1.60GHz，RAM4.00GB；软件环境为 Ubuntu 16.04.2 LTS，gcc version 5.4.0，qt version 5.6.2。

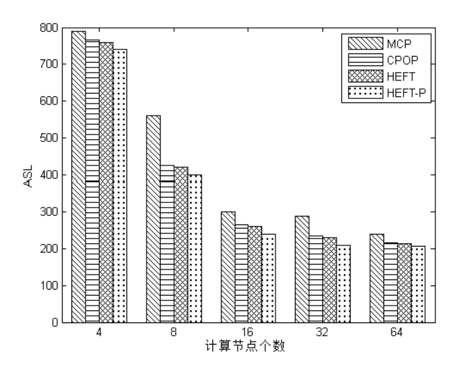
3.1　任务平均调度长度

图 4 横坐标代表计算节点的个数，计算节点的数目由 4 渐增到 64；纵坐标表示不同算法调度实际工作流 CyberShake 的 ASL，工作流的任务数量分别为 50 和 100。当计算节点个数较少时，本文算法与 HEFT，CPOP 算法的 ASL 相当，但随着计算节点数目的增加，本文策略始终能够产生最低的 ASL。

如图 4(a)所示，当任务数目小于计算节点数目时，各调度算法的 ASL 差异较小。由于此时系统可用资源相对丰富，各调度算法都能依据简单的启发式策略执行调度决策，影响 ASL 的主要因素成为可用资源总量，而非算法本身。在实验中发现，在模型剖分时将负载均衡阈值参数设为 0.2 时，本文策略调度结果表现最佳。如图 4(b)所示，随着计算节点数目的增加，HEFT-P 依然能够得到较好的调度结果。在 CyberShake 中，有一种叫做 Seismogram-Synthesis 的任务，它的实际平均数据读取量为 547MB/job 错误!未找到引用源。，但其所需的输入数据大小却仅为 150MB-250MB，这意味着存在一些输入数据被反复读取，造成了 I/O 资源的耗费与调度时间的延长。相较于其他算法，本策略能够取得较低的 ASL。



(a) 50 个 CyberShake 工作流的 ASL



(b) 100 个 CyberShake 工作流的 ASL

图 4　不同任务数目 CyberShake 工作流的 ASL

图 5 纵坐标表示不同算法调度 Montage 工作流的 ASL，工作流的任务数量分别为 25 和 100。结果显示，随着计算节点个数的增加，四种算法的 ASL 都有不同程度的下降，HEFT-P 始终能够产生最低的 ASL。根据实验结果发现，本策略对 CyberShake 工作流的调度效果优于对 Montage 工作流。因与 Montage 工作流相比，CyberShake 工作流的 CCR 值更高，说明本文策略更适用于数据密集型工作流的调度。

3.2　负载均衡对比

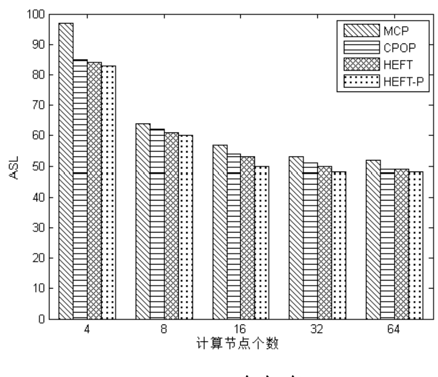
图 6 为当任务数目不同，计算节点相同情况下，不同算法的平均资源利用率。横坐标表示任务数目，任务数量由 200 增加到 1000；纵坐标表示不同算法的平均资源利用率。HEFT 算法在工作流调度时并没有考虑资源利用率的目标，由图可见，

[1]https://confluence.pegasus.isi.edu/display/pegasus/ WorkflowGenerator

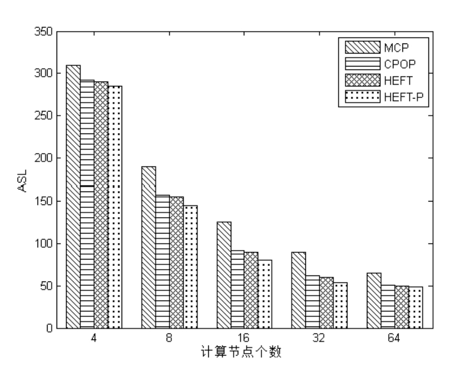
chinaXiv:201805.00481v1

ChinaXiv合作期刊
chinaXiv:201805.00481v1

随着任务数目的增多，HEFT-P 算法的资源利用率优于 HEFT 算法。



(a)　25 个任务



(b)　100 个任务

图 5　不同任务数目(a)25 (b)100 Montage 工作流的 ASL

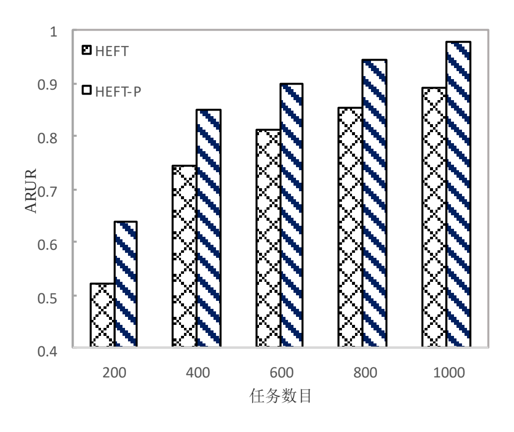


图 6　不同任务数目下算法的平均资源利用率

上述实验结果表明：云计算环境中，随着数据密集型应用任务规模的增大，任务间的依赖关系更加复杂，任务的通信开销及其对应的调度长度也在显著增加。而本文策略相比于 HEFT、CPOP 及 MCP 算法，能够较好地降低任务间的通信量，获得较优的调度效果。

4　结束语

本文通过对数据密集型应用在利用云平台进行执行处理时所遇到的挑战进行分析，提出了基于有向超图划分的数据密集型任务的 HEFT-P 调度算法。实验结果证明，本策略是一种有效的云计算环境下数据密集型工作流调度策略，相较于 HEFT、CPOP 及 MCP 算法，尤其是在任务量较大的情况下，能够在满足负载均衡的约束下获得更低的任务完成时间。在此基础上，接下来的研究方向是将该策略应用到真实的数据密集型应用中，对算法的可扩展性进行优化，对模型结构变换进行完善和扩展，通过研究分解操作来更好地对模型进行表达和处理。

参考文献：

[1] Chen C L P, Zhang Chunyang. Data-intensive applications, challenges, techniques and technologies: a survey on big data [J]. Information Sciences, 2014, 275 (11): 314−347.

[2] Armbrust M, Fox A, Griffith R, *et al.* A view of cloud computing [J]. International Journal of Computers & Technology, 2010, 53 (4): 50−58.

[3] 宫学庆，金澈清，王晓玲，等．数据密集型科学与工程：需求和挑战 [J]．计算机学报, 2012, 35 (8): 1563−1578. (Gong Xueqing, Jin Cheqing, Wang Xiaoling, *et al.* Data-Intensive Science and Engineering: Requirements and challenges [J]. Chinese Journal of Computers, 2012, 35 (8): 1563−1578.)

[4] Catalyurek U V, Boman E G, Devine K D, *et al.* Hypergraph-based dynamic load balancing for adaptive scientific computations [C]// Proc of Parallel and Distributed Processing Symposium. 2007: 1−11.

[5] Zhou D, Huang J, Schölkopf B. Learning with hypergraphs: Clustering, classification, and embedding [C]// Advances in Neural Information Processing Systems. 2006: 1601−1608.

[6] Zhao Han, Liu Xinxin, Li Xiaolin. Hypergraph-based task-bundle scheduling towards efficiency and fairness in heterogeneous distributed systems [C]// Proc of IEEE International Symposium on Parallel & Distributed Processing. 2010: 1−12.

[7] 郭文忠，陈国龙，彭少君．求解 VLSI 电路划分问题的混合粒子群优化算法 [J]．软件学报, 2011, 22 (5): 833−842. (Guo Wenzhong, Chen Guolong, Peng Shaojun. Hybrid particle swarm optimization algorithm for VLSI circuit partitioning [J]. Journal of Software, 2011, 22 (5): 833−842.)

[8] Ma Yonggang, Tan Guozhen, Wang Wei. A multi-objective hypergraph partitioning model for parallel computing [J]. International Journal of Parallel, Emergent and Distributed Systems, 2012, 27 (4): 337−346.

[9] Biswal P, Lee J R, Rao S. Eigenvalue bounds, spectral partitioning, and metrical deformations via flows [J]. Journal of the ACM, 2010, 57 (3): 13: 1−13: 23.

[10] Deveci M, Kaya K, Uçar B, *et al.* Hypergraph partitioning for multiple communication cost metrics: Model and methods [J]. Journal of Parallel & Distributed Computing, 2015, 77 (2015): 69-83.

[11] Devine K D, Boman E G, Heaphy R T, *et al.* Parallel hypergraph partitioning for scientific computing [C]// Proc of the 20th IEEE International Parallel & Distributed Processing Symposium. 2006: 10.

[12] 孙雪冬，徐晓飞，王刚．基于有向超图的资源约束下企业过程结构优化 [J]．软件学报, 2006, 17 (1): 59−67. (Sun Xuedong, Xu Xiaofei, Wang Gang. Directed hypergraph based and resource constrained enterprise process structure optimization [J]. Journal of Software, 2006, 17 (1): 59−68.)

ChinaXiv合作期刊

[13] Laura L, Nanni U, Temperini M. The organization of large-scale repositories of learning objects with directed hypergraphs [C]// Proc of International Conference on Web-Based Learning. Springer International Publishing. [S. l.] : Springer, 2014: 23−33.

[14] 肖鹏，胡志刚，屈喜龙．面向数据密集型工作流的能耗感知调度策略[J]．通信学报, 2015, 36 (1): 149−158. (Xiao Peng, Hu Zhigang, Qu Xilong. Energy-aware scheduling policy for data-intensive workflow [J]. Journal on Communications. 2015 (1) , 36 (1): 149-158.)

[15] 苑迎春，李小平，王茜，等．基于逆向分层的网格工作流调度算法 [J]. 计算机学报, 2008, 31 (2): 282-290. (Yuan Yingchun, Li Xiaoping, Wang Qian, *et al.* Bottom level based heuristic for workflow scheduling in grids [J]. Chinese Journal of Computers, 2008, 31 (2): 282-290.)

[16] Ahmad S G, Liew C S, Munir E U, *et al.* A hybrid genetic algorithm for optimization of scheduling workflow applications in heterogeneous computing systems [J]. Journal of Parallel & Distributed Computing, 2016, 87 (C): 80−90.

[17] Lin Bing, Guo Wenzhong, Xiong Naixue, *et al.* A pretreatment workflow scheduling approach for big data applications in multicloud environments [J]. IEEE Trans on Network & Service Management, 2016, 13 (3): 581-594.

[18] Mon E E, Thein M M, Aung M T. Clustering based on task dependency for data-intensive workflow scheduling optimization [C]// Many-Task Computing on Clouds, Grids, and Supercomputers. 2017: 20-25.

[19] Auffhammer M, Hsiang S M, Schlenker W, *et al.* Using weather data and climate model output in economic analyses of climate change [J]. Review of Environmental Economics & Policy, 2013, 7 (2): 181−198.

[20] Calheiros R N, Ranjan R, Beloglazov A, *et al.* CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms [J]. Software: Practice and Experience. Software Practice & Experience, 2010, 41 (1): 23–50.

[21] Bert A C, source, Norcott. IOzone [M]. [S. l.] : Chromo Publishing, 2012.

[22] Juve G, Chervenak A, Deelman E, *et al.* Characterizing and profiling scientific workflows [J]. Future Generation Computer Systems, 2013, 29 (3): 682−692.

chinaXiv:201805.00481v1